Homework 5 - Lists, Tuples, and Modules

CS 1301 - Intro to Computing - Spring 2022

Important

- Due Date: **Tuesday, February 22th, 11:59 PM**.
- This is an individual assignment. High-level collaboration is encouraged, **but your submission must be uniquely yours.**
- Resources:
  - TA Helpdesk
  - Email TA's or use class Piazza
  - How to Think Like a Computer Scientist
  - CS 1301 YouTube Channel
- Comment out or delete all function calls. Only import statements, global variables, and comments are okay to be outside of your functions.
- **Read the entire document before starting this assignment.**

The goal of this homework is for you to enhance your understanding of lists, tuples, and modules. The homework will consist of 5 functions for you to implement. You have been given the `HW05.py` skeleton file to fill out. However, below you will find more detailed information to complete your assignment. Read it thoroughly before you begin.

Hidden Test Cases: In an effort to encourage debugging and writing robust code, we will be including hidden test cases on Gradescope for some functions. You will not be able to see the input or output to these cases. Below is an example output from a failed hidden test case:

```
Test failed: False is not True
```

Written by Cynthia Medlej (cmedlej3@gatech.edu) & Audrey Cho (acho45@gatech.edu)

Helpful Information to Know

Modules

Modules allow you to use code from another source. This could include other code that you've written in a different Python file, or external code written by other programmers. To use a module, you must **import** it into your file first. There are multiple ways to do this. Let's look at some ways to import and use the `pi` constant and `sqrt` function from Python's built-in **math** module.

Note: By convention, `import` statements should be at the top of your file, **not in a function.**

- Importing the module itself:

```
import math

val = math.pi * math.sqrt(4)
```

- Importing specific item(s) from a module:

```
from math import pi, sqrt

val = pi * sqrt(4)
```

- Importing a module with a different name:

```
import math as m

val = m.pi * m.sqrt(4)
```

TV Show Night

Function Name: showToWatch()
Parameters: friendsFavShows (`list`), favoriteShow (`str`)
Returns: list of friends (`list`)
Description: You've been dying to watch one of your favorite shows but you don't want to watch it alone. Given your favoriteShow (`str`) and friendsFavShows (`list`), a list of tuples each containing your friends' names and a list of their favorite shows, write a function that returns a list of the friends that also want to watch your favoriteShow. If no one wants to watch your favoriteShow, your function should return the string `"Lonely night :("` . **The returned list should be in alphabetical order.**

Note: The list friendsFavShows will be in the format stated below:

```
[(friend1, [list of friend1's fav shows]),
 (friend2, [list of friend2's fav shows])...]
```

```
>>> friendsFavShows = [('Eric', ['Friends', 'B99']),
                       ('Anthony', ['Money Heist','Friends']),
                       ('Sara', ['Friends', 'Elite'])]
>>> favoriteShow = 'Friends'
>>> showToWatch(friendsFavShows, favoriteShow)
['Anthony', 'Eric' , 'Sara']
```

```
>>> friendsFavShows = [('Maria', ['Euphoria']),
                       ('Romy', ['Gilmore Girls']),
                       ('Lynn', ['You', 'Gossip Girl'])]
>>> favoriteShow = 'Euphoria'
>>> showToWatch(friendsFavShows, favoriteShow)
['Maria']
```

Label Fixing

Function Name: fixLabels()
Parameters: labelList (`list`)
Returns: list of correct labels (`list`)
Description: While working at a store, you are assigned the task of labeling items with their names and prices. However, you notice that the item name labels and price labels got mixed together in one stack! You know that the order of the name labels correctly corresponds to the order of the price labels as they appear in the stack. Write a function that takes in a list containing names (`str`) and prices (`float`) of items and returns an organized list of tuples that match the item name to its price. **The returned list should be in alphabetical order.**

Note: If the number of name labels does not match the number of price labels, return `'Missing labels'` .

```
>>> labelList = [2.00, 'apples', 0.99, 'bananas']
>>> fixLabels(labelList)
[('apples', 2.00), ('bananas', 0.99)]
```

```
>>> labelList = [2.99, 5.49, 'chocolate', 1.99, 'ice cream', 'candy']
>>> fixLabels(labelList)
[('candy', 1.99), ('chocolate', 2.99), ('ice cream', 5.49)]
```

New Spotify Playlist

Function Name: newPlaylist()
Parameters: playlist (`list`)
Returns: list of songs (`list`)
Description: You want to spice up your Spotify playlist, so you ask your friend for recommendations. Given a list of tuples containing the name of the song with their length in the following format `('name', 'minutes:seconds')` , write a function that returns a list containing a tuple of all the songs in the list and the total time of the playlist in **minutes** (round your answer to two decimals). **The tuple in the returned list should be sorted in alphabetical order.**

```
>>> playlist = [("Mr/Mme", "6:22"),
                ("Les yeux de la Mama", "3:04"),
                ("Elle me dit", "3:34")]
>>> newPlaylist(playlist)
[('Elle me dit', 'Les yeux de la Mama', 'Mr/Mme'), 13.0]
```

```
>>> playlist = [("All Too Well", "10:13"),
                ("Forever & Always", "3:46"),
                ("Love Story", "3:56"),
                ("Mr. Perfectly Fine", "4:38")]
>>> newPlaylist(playlist)
[('All Too Well', 'Forever & Always', 'Love Story', 'Mr. Perfectly Fine'), 22.55]
```

Birthday Planner

Function Name: birthdays()
Parameters: friends (`list`), birthdates (`list`)
Returns: list of names (`list`)
Description: Your friends' birthdays are coming up! However, your schedule for 2022 is packed and you can't make it on weekdays. A weekday is any day from Monday to Friday. Write a function that takes in two lists as parameters: one containing the names of your friends and another containing tuples of their birthdays in the format: `(month, day)`. Using the calendar module, your function should return a list containing the name of the friends whose birthdays fall on weekends.**The returned list should be in alphabetical order.**

Note: Use the **weekday()** function from the **calendar module** to check the number of the day of the week.

```
>>> friends = ['Audrey', 'Paige', 'Anastasia', 'Ramya', 'Lasya', 'Cynthia']
>>> birthdates = [(5,6), (4,7), (1,19), (12,4), (9,10), (3,6)]
>>> birthdays(friends, birthdates)
['Cynthia', 'Lasya', 'Ramya']
```

```
>>> friends = ['Sarra', 'Isabelle', 'Andrea']
>>> birthdates = [(12,31), (4,9), (4,8)]
>>> birthdays(friends, birthdates)
['Isabelle', 'Sarra']
```

Super Smash Bros.

Function Name: smashBros()
Parameters: fighterList (`list`), opponent (`str`)
Returns: list of good picks (`list`)
Description: You and your friend are playing a game of Super Smash Bros and you want to know if any of your fighters is a good counter pick against your friend's fighter. Write a function that takes in two parameters: a list of your fighters (`list`) and your friend's fighter (`str`). If any of the fighters in your list are good counter picks, return a list containing the names of those fighters. If not, return the string `'No counters!'` .**The returned list should be in alphabetical order.**

We have provided you with a file called **`smashData.py`** that contains a function called `counterPick()` . This function will take in a fighter name and return a list of the good counter picks to that fighter. If the fighter is not found, the function returns `None` . **You should not submit `smashData.py` on Gradescope.**

```
>>> fighterList = ['bowser', 'marth']
>>> smashBros(fighterList, 'ness')
['bowser']
```

```
>>> fighterList = ['mewtwo', 'kirby', 'link', 'pikachu']
>>> smashBros(fighterList, 'mewtwo')
['kirby', 'pikachu']
```

Grading Rubric

Function	Points
showToWatch()	20
fixLabels()	20
newPlaylist()	20
birthdays()	20
smashBros()	20
Total	**100**

Provided

The `HW05.py` skeleton file has been provided to you. This is the file you will edit and implement. All instructions for what the functions should do are in this skeleton and this document. The module `smashData.py` has been provided to you as well and is located in the HW05 folder on Cnvas.

Submission Process

For this homework, we will be using Gradescope for submissions and automatic grading. When you submit your `HW05.py` file to the appropriate assignment on Gradescope, the autograder will run automatically. The grade you see on Gradescope will be the grade you get, unless your grading TA sees signs of you trying to defeat the system in your code. You can re-submit this assignment an unlimited number of times until the deadline; just click the “Resubmit” button at the lower right-hand corner of Gradescope. You do not need to submit your `HW05.py` on Canvas. You also do not need to submit the provided module `smashData.py` on Gradecope.